AMENDMENTS

In the Claims:

This listing of claims replaces all prior versions and listings of claims in the application.

1. (Currently Amended) A method for identifying infected program instructions, comprising the steps of:

inserting a dynamic execution layer interface (DELI) between computing device hardware and the program instructions;

monitoring the program instructions as they enter the DELI to determine if the ~~code has~~ program instructions have been previously ~~processed by the computing device hardware~~ cached within the DELI, wherein the determination of whether the program instructions have been cached is responsive to an association between native application code and one or more analogues that have been transformed within the DELI; and when it is the case that the ~~application code has~~ program instructions have not been previously ~~processed~~ cached within the DELI,

analyzing the program instructions to determine if the program instructions are infected.

2. (Original) The method of claim 1, wherein the step of analyzing the program instructions comprises an investigation of the contents of instructions within code fragments.

3. (Original) The method of claim 1, wherein the step of analyzing the program instructions comprises inserting decrypted program instructions into a virus detection manager.

4. (Original) The method of claim 3, further comprising the step of:

releasing program instructions from the virus detection manager when infected program instructions are not detected.

5. (Original) The method of claim 3, wherein the step of analyzing the program instructions comprises performing a signature comparison with the contents of the code fragments.

6. (Original) The method of claim 3, wherein the step of analyzing the program instructions comprises monitoring the behavior of the contents of the code fragments in a virtual computing device.

7. (Original) The method of claim 3, wherein the step of analyzing the program instructions comprises applying a plurality of tests on the contents of the code fragments in a virtual computing device.

8. (Original) The method of claim 4, further comprising the step of:
processing the released program instructions in computer hardware.

9. (Currently Amended) A system for detecting infected program instructions in active software applications, comprising:
means for intercepting program instructions designated for execution within a computing device;
means for transforming the program instructions;
means for determining when the intercepted program instructions have not been processed by the computing device responsive to an association between native application code from the active software applications and one or more analogues that have been cached within a dynamic execution layer inserted between a processor and program instructions; and
means for analyzing the intercepted program instructions that have not been processed by the computing device prior to forwarding the intercepted program instructions to computer hardware.

10. (Original) The system of claim 9, further comprising:
means for gaining control over execution of program instructions.

11. (Original) The system of claim 9, further comprising:
means for executing program instructions.

12. (Original) The system of claim 9, wherein the means for intercepting comprises a dynamic execution layer interface (DELI).

13. (Original) The system of claim 9, wherein the means for analyzing the intercepted program instructions comprises a virus detection manager.

14. (Original) The system of claim 13, wherein the virus detection manager comprises a controller configured to apply a plurality of virus detection tests over the contents of the intercepted program instructions.

15. (Currently Amended) A virus detection program stored on a computer-readable medium, comprising:
logic configured to intercept program instructions;
logic configured to transform the program instructions;
logic configured to determine if the intercepted program instructions have not been processed by a computing device responsive to an association between application code and one or more analogues that have been cached within a dynamic execution layer inserted between a processor and program instructions; and
logic configured to determine when the intercepted program instructions that have not been processed by the computing device are infected with a virus.

16. (Original) The program of claim 15, further comprising:
logic configured to gain control over execution of intercepted program instructions.

17. (Original) The program of claim 15, further comprising:
logic configured to execute program instructions.

18. (Original) The program of claim 15, further comprising:

logic configured to forward non-infected intercepted program instructions to the computing device.

19. (Original) A computer system, comprising:

a processor;

an execution memory;

a dynamic execution layer interface (DELI) residing between at least one application and the processor, wherein the DELI comprises:

a core configured to cache and execute certain application code fragments;

an application programming interface configured to provide access to caching and executing functions of the core to a virus detection manager; and

a system control and configuration layer configured to provide policies for operation of the core.

20. (Original) The system of claim 19, wherein the virus detection manager is configured to apply at least one virus detection test on the contents of application code fragments.

21. (Original) The system of claim 19, wherein the core is configured to process executable application code fragments from the at least one application that have not been previously sent to the processor.

22. (Original) The system of claim 21, wherein the virus detection manager controls whether application code fragments are released to the processor.

23. (Original) The system of claim 22, wherein application code fragments that contain at least one virus signature are not released to the processor.

24. (Original) The system of claim 22, wherein application code fragments that behave in a manner consistent with known virus attacks are not released to the processor.